Microprocessors B (17.384)

Spring 2011

Lecture Outline

Class # 02

February 01, 2011

Dohn Bowden

Today's Lecture

- Administrative
- Microcontroller Hardware and/or Interface
- Programming/Software
- Lab
- Homework

Course Admin

Administrative

- Admin for tonight ...
  - Syllabus Highlights
    - Lab #1 is in two weeks (February 15th)
  - We shall perform Lab #1 next week ... NO lecture

Syllabus Review

Week	Date	Topics	Lab	Lab Report Due
~~1~~	~~01/25/11~~	~~PIC pin out, C programming, Watchdog Timer, Sleep~~		
2	02/01/11	General-purpose IO, LED/switch IO, FSM	1	
3	02/08/11	Lab	1 con't	
4	02/15/11	Interrupts, Timers, interrupt-driven IO	2	1
5	02/22/11	Lab	2 con't	
6	03/01/11	Asynchronous and Synchronous Serial IO (UART, I^2C, SPI)	3	2
7	03/08/11	**Examination 1**		
X	03/15/11	**No Class – Spring Break**		
8	03/22/11	Lab	3 con't	
9	03/29/11	Serial EEPROM operation, DAC, DC motor control, Servos, Stepper motor control	4	3
10	04/05/11	Lab	4 con't	
11	04/12/11	Advanced Hardware Topics	Project	4
12	04/19/11	**Examination 2**		
13	04/26/11	Work on Course Project	Project	
14	05/03/11	Final Exam/Course Project Brief and Demonstration	Demo	
				5

Access to Labs

- To gain access to BL-407
  - You will need to have an access cards
  - Access cards are given out on the South Campus at Access Services
    - Arrangements are done through the Continuing Education Office
  - I need your UMS# (will be on your Access Card) for room access

Course Web Site

- The Course Web site Homepage is at:

  http://faculty.uml.edu/dbowden

- Any issues?

Email Distribution List

- For those who have sent me your email address or addresses at ...

  » Dohn_Bowden@uml.edu

- I sent a test email to you

Chat Page

- Still looking into what is available through the school
  - More to follow as it develops

Microcontroller Hardware and / or Interfaces

Template Sheet

Chapter 8 ...

continued ...

Parallel Port Operation

General Purpose Input/Output (I/O)

- The simplest type of I/O via the PIC24 external pins are ...
  - **Parallel I/O (PIO) ports**
- A PIC24 family can have multiple PIO ports named ...
  - PORTA, PORTB, PORTC, PORTD, etc
- Each is 16-bits, and the number of PIO pins depends on the particular PIC24 and package.

General Purpose Input/Output (I/O)

- The PIC24HJ32GP202/28 pin package has ...
  - PORTA
    - Bits ... RA4 through RA0
  - PORTB
    - Bits ... RB15 through RB0
- Generically referred to as ... PORT*x*

General Purpose Input/Output (I/O)

- Each pin on PORT*x* can either be ...
  - An input ... or ...
  - An output
- Data direction is controlled by ...
  - The corresponding bit in the ...
    - TRIS*x* registers
      - '1' = input
      - '0' = output
- The LAT*x register holds the last value written to PORTx*

PORT B example …

PORT B Example

Set the upper 8 bits of PORTB to outputs, lower 8 bits to be inputs:

```
TRISB = 0x00FF;
```

Drive RB15, RB13 high;
others low:

```
PORTB = 0xA000;
```

Wait until input RB0 is high:

```
while ((PORTB & 0x0001) == 0);
```

Test returns true while RB0=0
so loop exited when RB0=1

Wait until input RB3 is low:

```
while ((PORTB & 0x0008) == 1);
```

Test returns true while RB3=1
so loop exited when RB3=0

PORT B Example … continued

Individual PORT bits are named as _RB0, _RB1, .._RA0, etc. so this can be used in C code.

Wait until input RB2 is high:

```
while (_RB2 == 0);
```

Test returns true while RB2=0 so loop exited when RB2=1. Can also be written as:

```
while (!_RB2);
```

Wait until input RB3 is low:

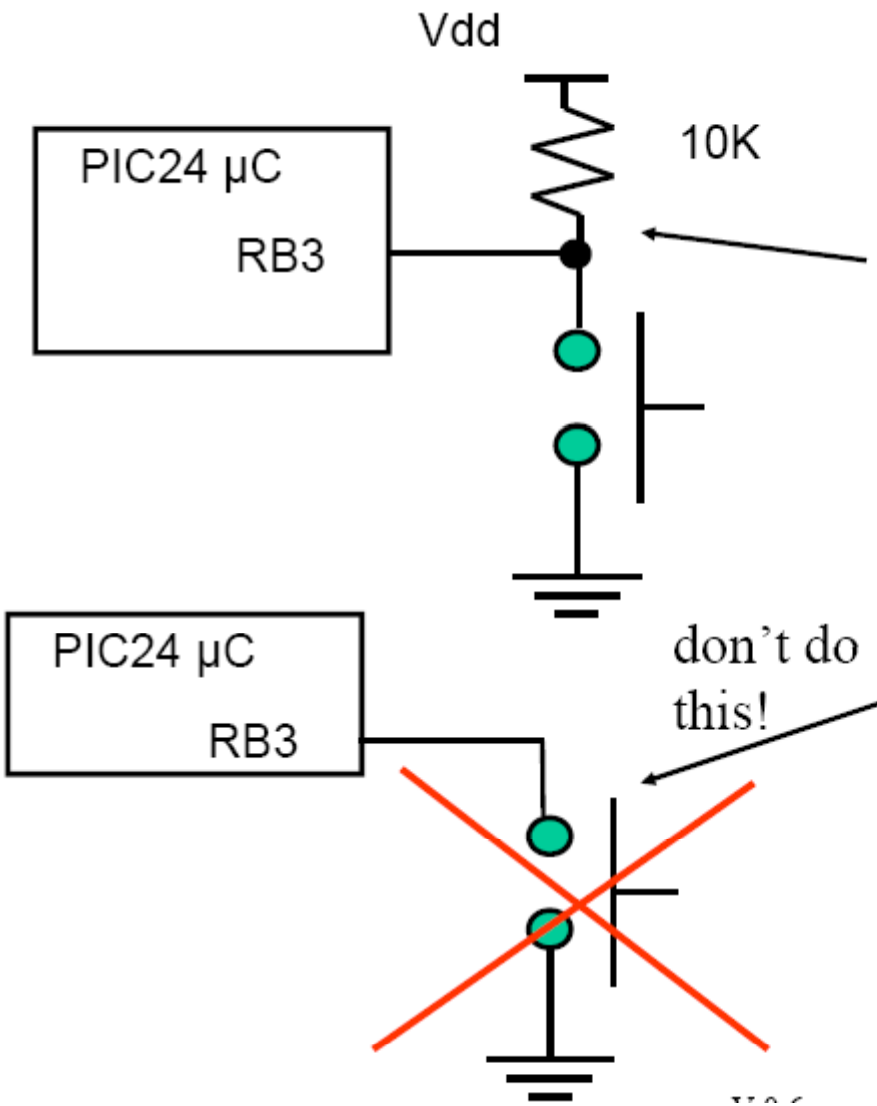
```
while (_RB3 == 1) ;
```

Test returns true while RB3=1 so loop exited when RB3=0 Can also be written as:

```
while (_RB3);
```

Switches …

Switch Input



External pullup

When switch is pressed RB3 reads as '0', else reads as '1'.

If pullup not present, then input would float when switch is not pressed, and input value may read as '0' or '1' because of system noise.

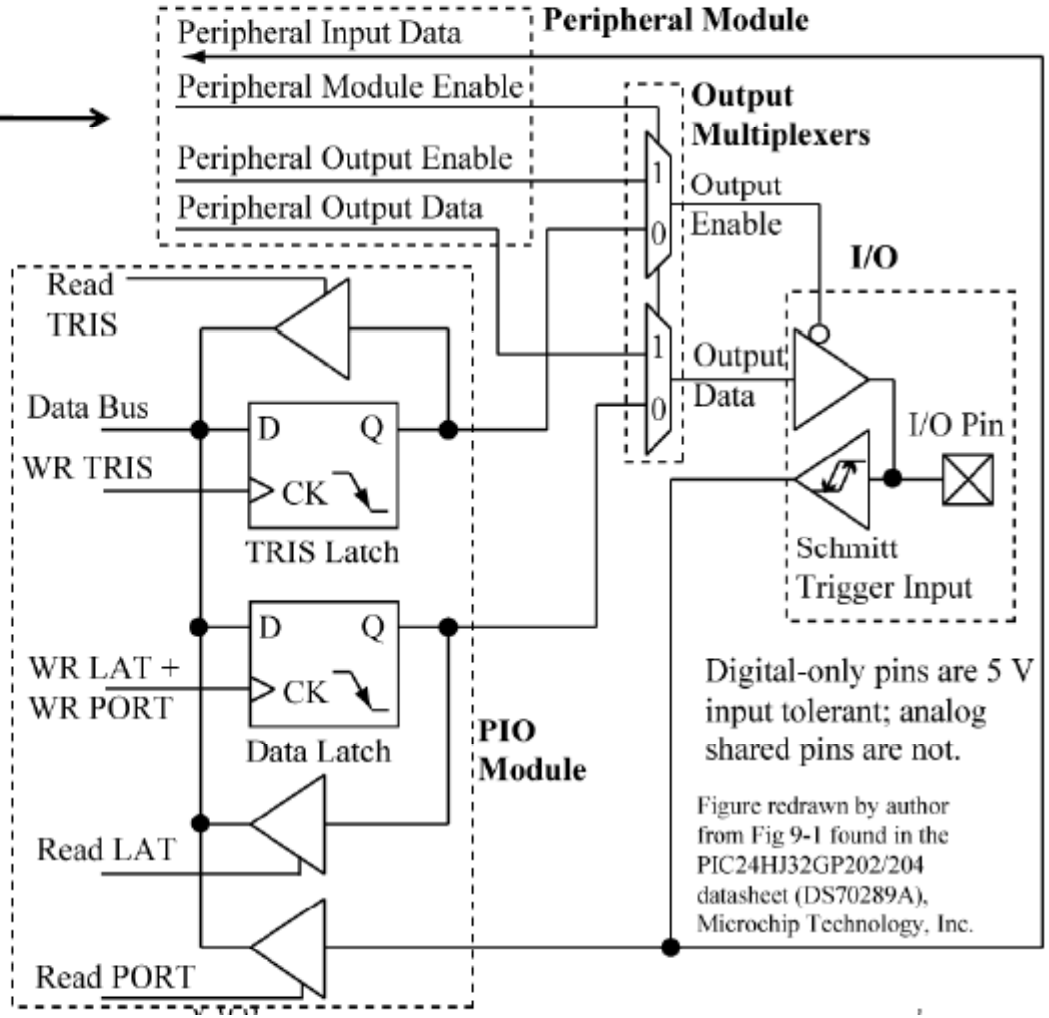
PORT*x* and LAT*x* …

PORTx Pin Diagram

External pin shared with other on-chip modules

TRIS bit controls tristate control on output driver

Reading LATx reads last value written; reading PORTx reads the actual pin



LAT*x* versus PORT*x*

- Writing LAT*x* is the same as ...
  - Writing PORT*x*
- Both writes go to the latch
- Reading LAT*x* ...
  - Reads the latch output ...
    - The last value written
- Reading PORT*x* ...
  - Reads the actual pin value

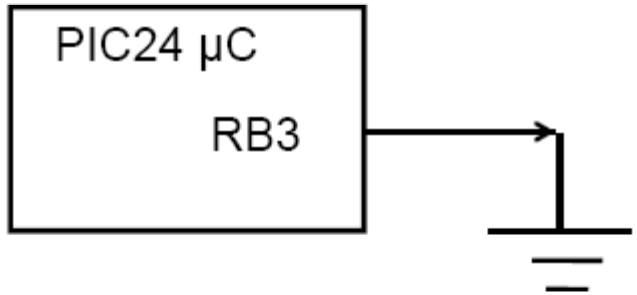
LAT*x* versus PORT*x* … continued

- Configure RB3 as an open-drain output … then write a ‘1’ to it

  – Open-drain … means that the PMOS pull-up is disabled
    - Removing the ability of the port pin to drive it’s output high
      – Output pin floats when driven high

- The physical pin is tied to ground, so it can never go high

- Reading _RB3 returns a ‘0’ … but reading _LATB3 returns a ‘1’ … the last value written



LAT*x* versus PORT*x* ... continued

- If we have the following two “C” statements consecutively ...

```
_RB2 = 1;
_RB3 = 1;
```

- An incorrect assignment for the second assignment can occur
  - Per the datasheet ...
    - Depends on external pin loading ... and ...
    - Internal clock speed

- x

LAT*x* versus PORT*x* ... continued

- Therefore use the following two “C” statements vice previous ...

```
_LATB2 = 1;
_LATB3 = 1;
```

- We therefore will use ...
  - LAT*xy* ...
    - For port writes
  - And ... RB*xy* ...
    - For reading a pin state

LAT*x* versus PORT*x* … continued

- Why does the previous occur?
  - When the compiler coverts from “C” to assembly

```
_LATB0 = 1;
_LATB1 = 1;
```

Compiler →

```
bset LATB,#0
bset LATB,#1
```

```
_RB0 = 1;
_RB1 = 1;
```

Compiler →

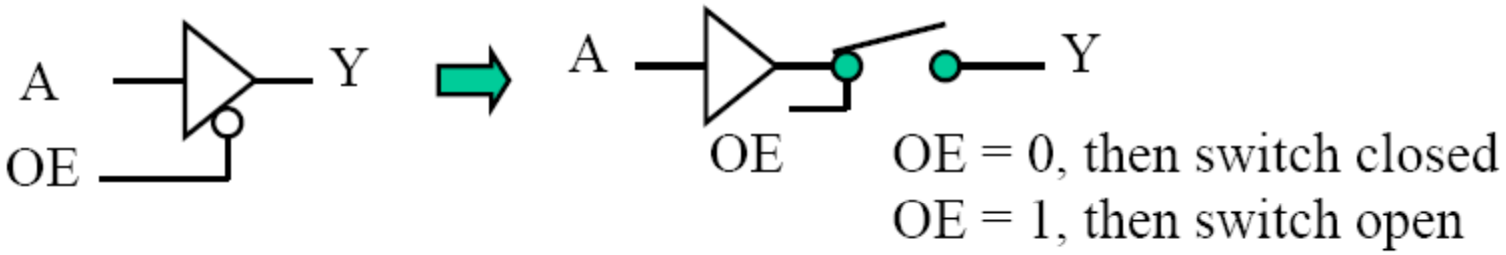
```
bset PORTB,#0
bset PORTB,#1
```

TRI-State Buffer …

Tri-State Buffer (TSB) Review

- A tri-state buffer (TSB) has ...
  - Input ...
  - Output ... and ...
  - Output enable (OE) pins

- Output can either be ...
  - '1' ...
  - '0' ... or ...
  - 'Z' (high impedance)

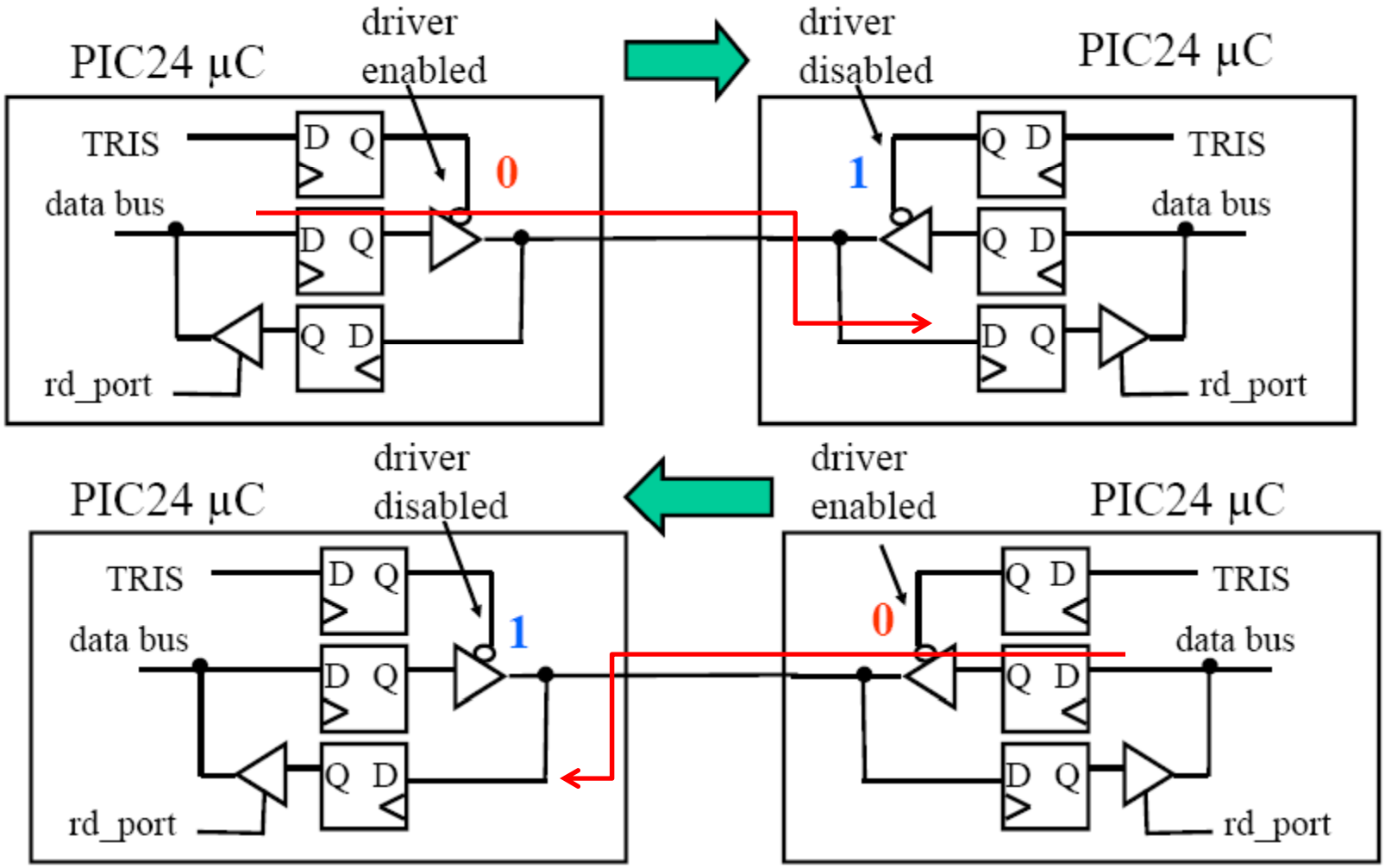


Bidirectional Data Link …

Bidirectional Data Link

- Uses one wire
- Data is flowing either from ...
  - CPU_a to CPU_b ... or ...
  - CPU_b to CPU_a
- But ... NEVER ... both directions at the same time ...
  - Can cause an indeterminate voltage and programming error

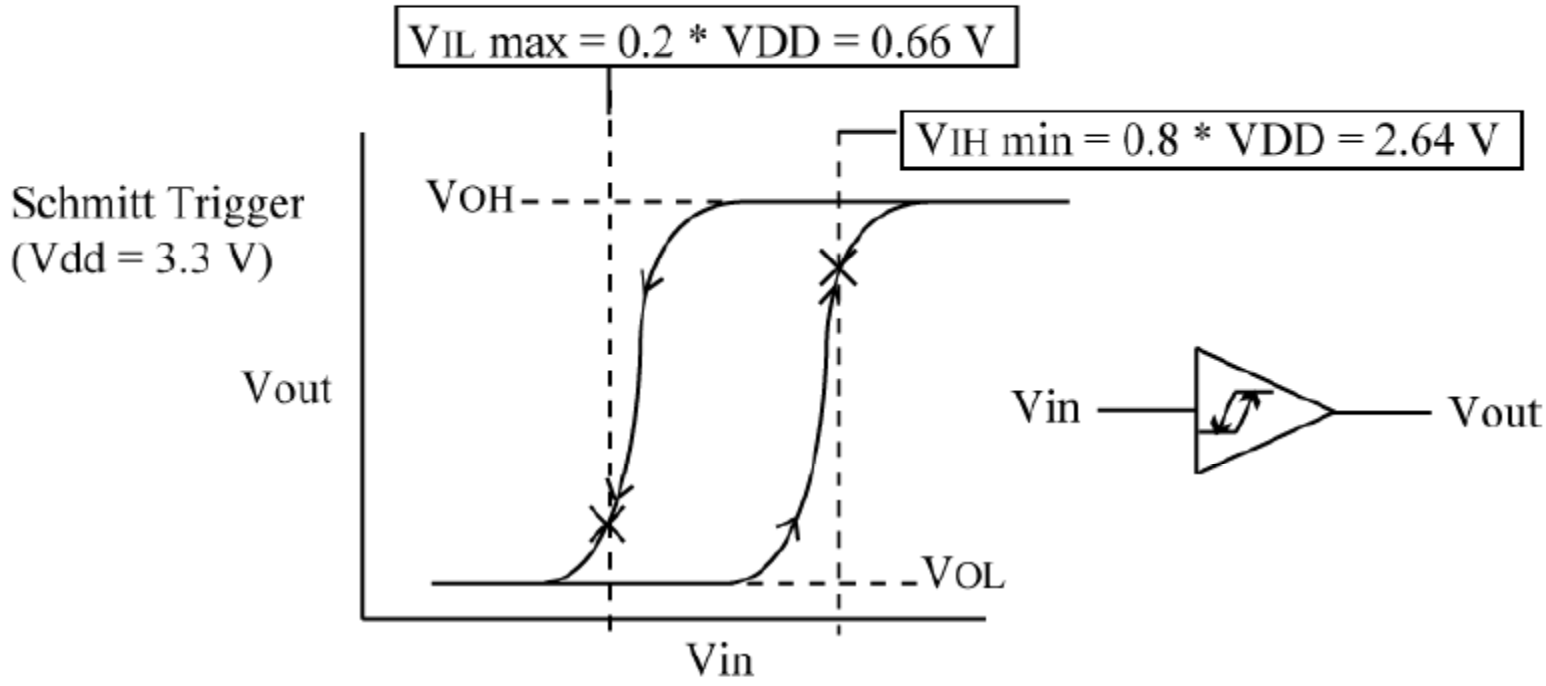
Bidirectional Data Link



Schmitt Trigger Input Buffer …

Schmitt Trigger Input Buffer

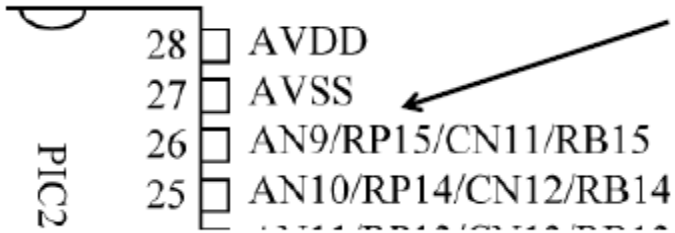
- Each PIO input has a *Schmitt trigger* input buffer ...
  - This transforms slowly rising/falling input transitions into ...
    - Sharp rising/falling transitions internally



PORT*x* ...

PORTx Shared Pin Functions

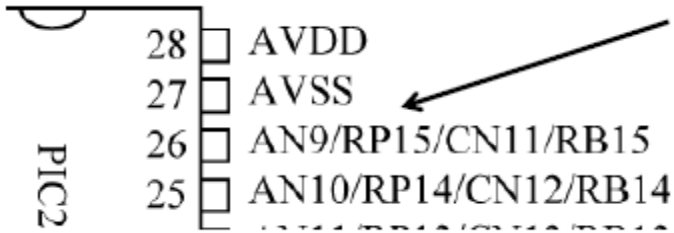
- External pins are shared with other on-chip modules
- Just setting _TRISx = 1 may be not be enough to configure a PORT*x* pin as an input
  - Depending on what other modules share the pin ...



PORTx Shared Pin Functions

- RB15 is shared with ... AN9
  - Which is an analog input to the on-chip Analog-to-Digital Converter (ADC)

- We must disable the Analog-to-Digital Converter



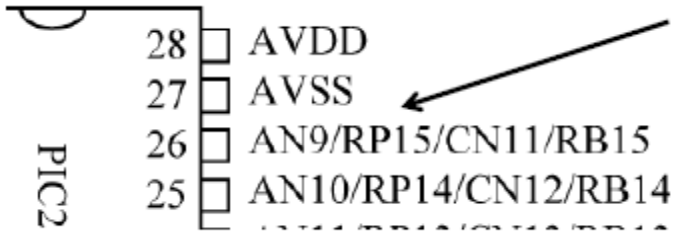
PORTx Shared Pin Functions

```
_PCFG9 = 1;
```
← Disables analog function

```
_TRISB15 = 1;
```
← Configure as input

```
_PCFG9 = 1;
```
← Disables analog function

```
_TRISB15 = 0;
```
← Configure as output



PORTx Shared Pin Functions

- _PCFG9 ... controls AN9 which shares same pin as RB15
  - Register 18-7 (datasheet)
    - “1” --- Digital mode
    - “0” --- Analog mode

```
_PCFG9 = 1;   <-------  Disables analog function
_TRISB15 = 1; <-------  Configure as input
```

```
_PCFG9 = 1;   <-------  Disables analog function
_TRISB15 = 0; <-------  Configure as output
```

Analog/Digital Pin versus Digital-only Pin …

Analog/Digital Pin versus Digital-only Pin

- Pins with shared analog/digital functions have a maximum input voltage of ...
    - Vdd + 0.3 V ... therefore ... 3.6 V
- Pins with no analog functions (“digital-only” pins) are ...
    - 5 V tolerant ... their maximum input voltage is ... 5.6 V
        - Can receive digital inputs from 5V components

Analog/Digital Pin versus Digital-only Pin

- Most PIO pins can only source or sink a maximum of ...
    - 4 mA
- You may damage the output pin if you ...
    - Tie a load that tries to sink/source more than 4 mA

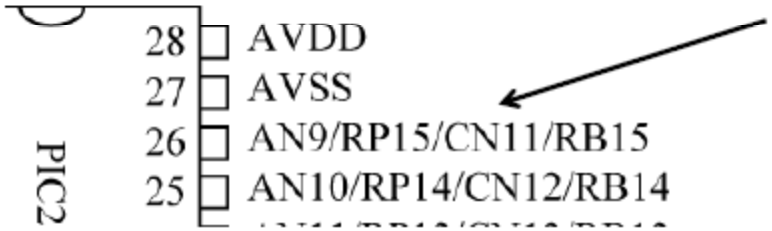
Internal Weak Pull-ups …

Internal Weak Pull-ups

- CN pins ...
  - Input Change Notification interrupt pins ...
    - Will be discussed in detail later in the semester

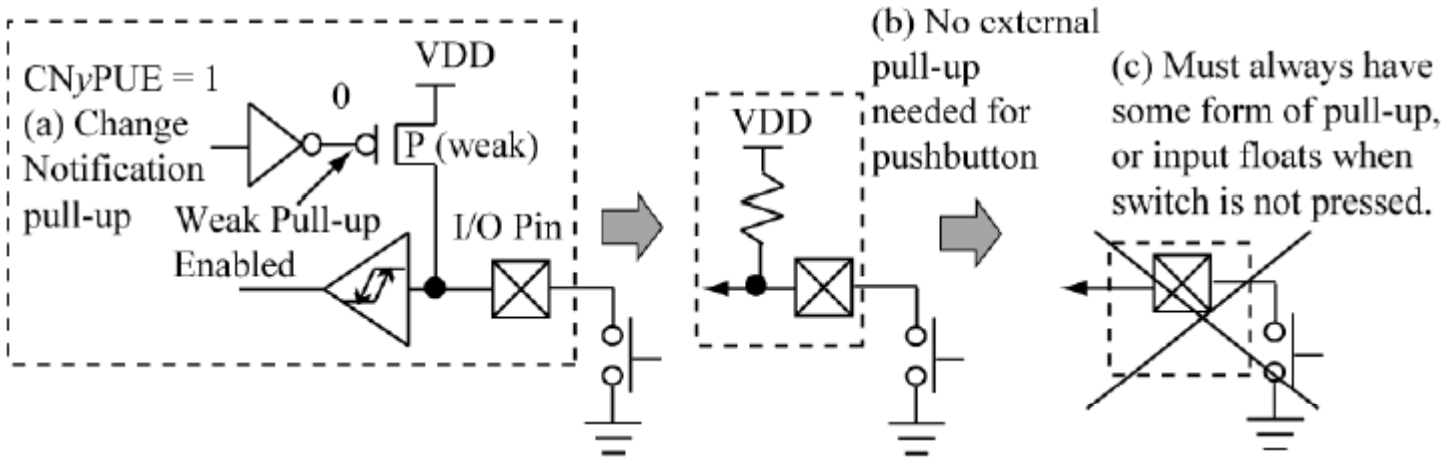
Internal Weak Pull-ups

- External pins with a CN*y* pin function have ...
  - A weak internal pull-up that can be enabled or disabled
- The weak pull-up is useful for eliminating the need for an ...
  - External pull-up resistor



Internal Weak Pull-ups

- For example ... when connecting a switch



- Change notification input ... to enable pull-up ...

  » CN11PUE = 1;

- To disable pull-up ...

  » CN11PUE = 0;

Internal Weak Pull-ups

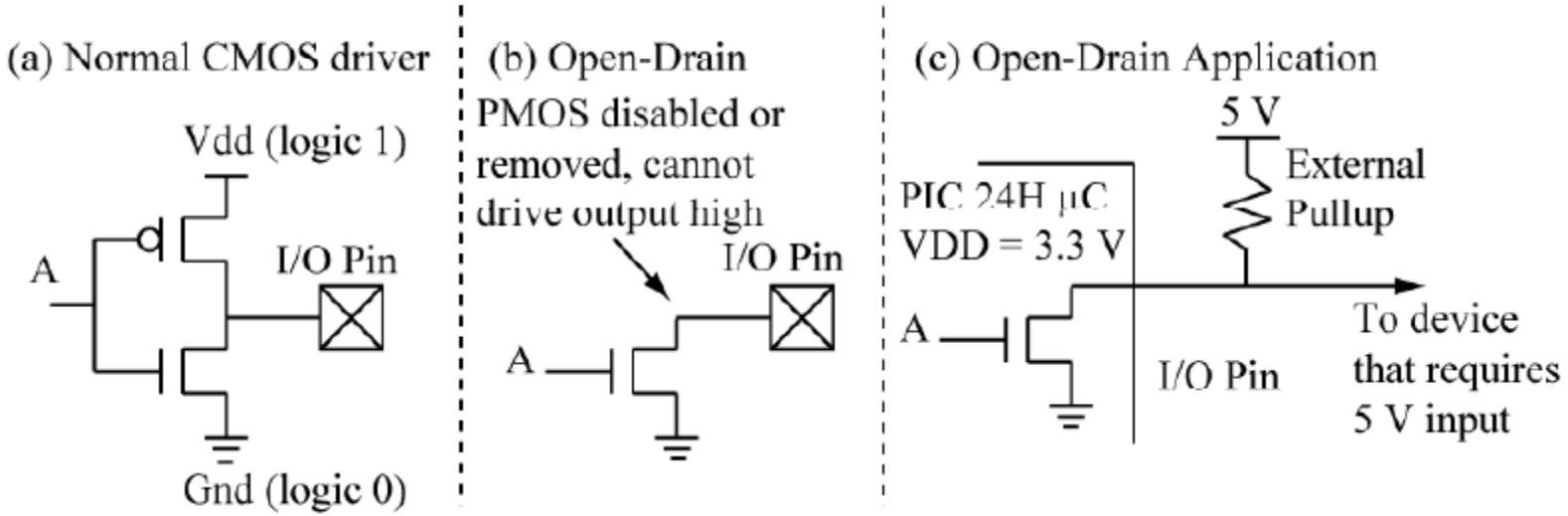
- CN11PUE ...
  - Datasheet ...Table 4-2

Open Drain Outputs …

Open Drain Outputs

- Open-drain ... means that the PMOS pull-up is disabled
  - Removing the ability of the port pin to drive it's output high
    - Output pin floats when driven high
- Each PIO pin can be configured as an *open drain output ...*
  - Which means the pull-up transistor is disabled

Open Drain Outputs



_ODC*xy* = 1 enables open drain, _ODC*xy* = 0 disables open drain

```
_ODCB15 = 1;
```
← Enables open drain on RB15

Port Configuration Macros …

Port Configuration Macros

- For convenience, we supply macros/inline functions that hide pin configuration details ...

```
CONFIG_RB15_AS_DIG_OUTPUT();

CONFIG_RB15_AS_DIG_INPUT();
```

Port Configuration Macros

- These macros are supplied for each port pin
- These functions change depending on the particular PIC24 ... therefore the ...
  - *include/devices* directory
    - Has a include file for each PIC24 ... and ... the correct file is included by the *include/pic24_ports.h* file
- **NOTE** ... the *include/devices* directory is the directory of the material supplied with the textbook, not the compiler

Other Port Configuration Macros

- Other macros are provided for pull-up and open drain configurations ...

```
ENABLE_RB15_PULLUP();
DISABLE_RB15_PULLUP();
ENABLE_RB13_OPENDRAIN();
DISABLE_RB13_OPENDRAIN();
CONFIG_RB8_AS_DIG_OD_OUTPUT();
```

Output + Open drain config in one macro

Other Port Configuration Macros

- General forms are ...
  - » ENABLE_R*xy*_PULLUP()
  - » DISABLE R*xy* PULLUP() ENABLE *Rxy* OPENDRAIN()
  - » DISABLE_R*xy*_PULLUP()
  - » *ENABLE_Rxy*_OPENDRAIN()
  - » DISABLE_R*xy*_OPENDRAIN()
  - » CONFIG_Rxy_AS_DIG_OD_OUTPUT()

- A port may not have a pull-up if it does not share the pin with a change notification input ...
  - In this case the macro does not exist and you will get an error compile the code message when you try to code

Configuration Macros

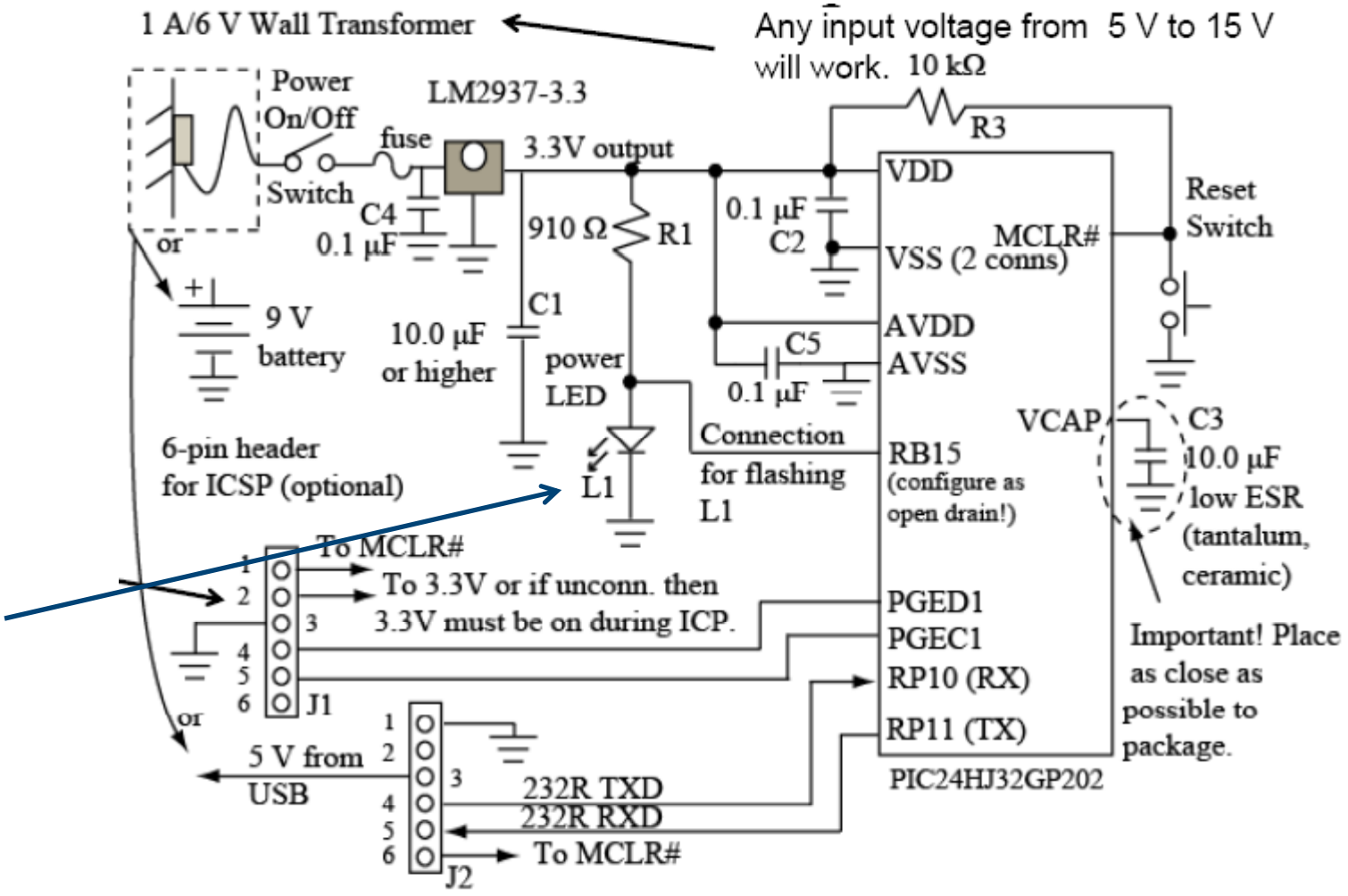
- Configuration Macros ...
  - You may want to initially write your code without using macros ...
    - To become familiar with how to perform the actual functions

Programming/Software

Flashing LED1…

Flashing LED1

- **Our initial hardware/software problem will be to flash LED1**



Flashing LED1

- The textbook provides a solution to the problem of flashing LED1
  - It provides advanced programming techniques ... which ...
    - Will reduce code ... but ...
      - Adds complexity for the novice “C” programmer
- I recommend ...
  - Reviewing the Code to understand what is being performed
  - Write the code to your knowledge level

Flashing LED1

- We can ...
  - Utilize the provided code ... or ...
  - Utilize the methods from last semester ... or ...
  - A combination of both ...
    - Take the best from each
      - And ... what you understand!

LEDflash.c …

LEDflash.c

```
#include "pic24_all.h"

/**
A simple program that
flashes an LED.
*/

#define CONFIG_LED1() CONFIG_RB15_AS_DIG_OD_OUTPUT()

#define LED1 _LATB15

int main(void) {

  configClock();     //clock configuration
  /********** PIO config **********/
  CONFIG_LED1();   //config PIO for LED1
  LED1 = 0;

  while (1) {
    DELAY_MS(250);  //delay
    LED1 = !LED1;   // Toggle LED
  } // end while (1)
}
```

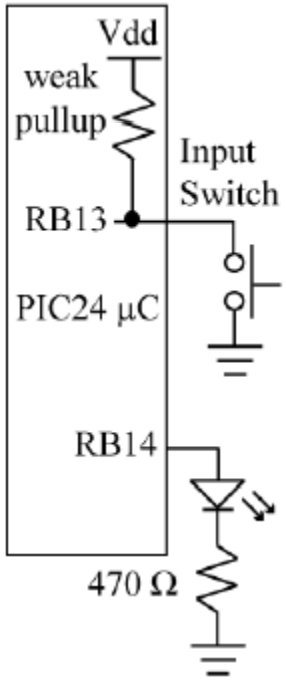
Defined in device-specific header file in *include\devices* directory in the book source distribution.
Macro `CONFIG_RB15_AS_DIG_OD_OUTPUT()` contains the statements `_TRISB15=0, _ODCB15 = 1`

LED1 macro makes changing of LED1 pin assignment easier, also improves code clarity.

`DELAY_MS(ms)` macro is defined in *common\pic24_delay.c* in the book source distribution, `ms` is a `uint32` value.

LED/Switch IO: Count number of press/releases

```
/// LED1, SW1 Configuration
#define CONFIG_LED1()  CONFIG_RB14_AS_DIG_OUTPUT()
#define LED1   _LATB14      //led1 state
inline void CONFIG_SW1()  {
  CONFIG_RB13_AS_DIG_INPUT();    //use RB13 for switch input
  ENABLE_RB13_PULLUP();          //enable the pull-up
}
#define SW1                  _RB13   //switch state
#define SW1_PRESSED()    SW1==0   //switch test
#define SW1_RELEASED()   SW1==1   //switch test
```



Count number of switch presses.

```
main(){
 ...other config...
 CONFIG_SW1();
 DELAY_US(1);
 CONFIG_LED1();
 LED1 = 0;
 while (1) {
   if (SW1_PRESSED()) {
   //switch pressed
   //toggle LED1
   LED1 = !LED1
   }
 }
}
```

a. Incorrect, LED1 is toggled as long as the switch is pushed, which could be a long time!

```
main(){
 ...other config...
 CONFIG_SW1();
 DELAY_US(1); //pull-up delay
 CONFIG_LED1();
 LED1 = 0;
 while (1) {
  // wait for press, loop(1)
  while (SW1_RELEASED());
  DELAY_MS(15); //debounce
  // wait for release, loop(2)
  while (SW1_PRESSED());
  DELAY_MS(15); // debounce
  LED1 = !LED1; //toggle LED
 }
```

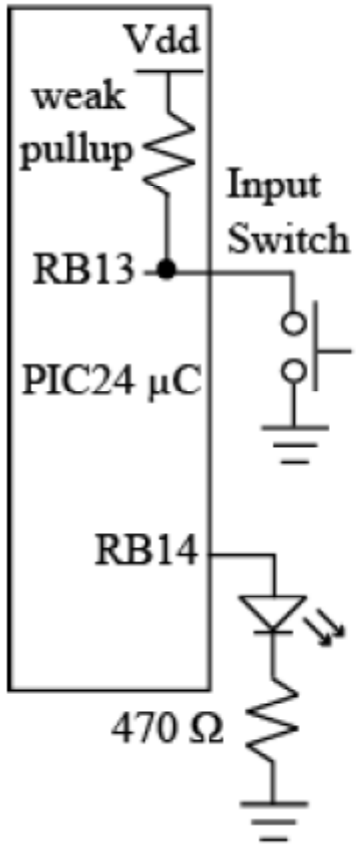
b. Correct, loop(1) executed while switch is not pressed. Once pressed, code becomes trapped in loop(2) until the switch is released, at which point LED1 is toggled.

I/O Configuration

- Use macros to isolate pin assignments for physical devices so ...
  - It is easy to change code if (WHEN!) the pin assignments change!

```
/// LED1, SW1 Configuration
#define CONFIG_LED1()  CONFIG_RB14_AS_DIG_OUTPUT()
#define LED1   _LATB14      //led1 state
inline void CONFIG_SW1()  {
  CONFIG_RB13_AS_DIG_INPUT();   //use RB13 for switch input
  ENABLE_RB13_PULLUP();         //enable the pullup
}
#define SW1                _RB13  //switch state
#define SW1_PRESSED()    SW1==0   //switch test
#define SW1_RELEASED()   SW1==1   //switch test
```

Counting # of Press/Releases



Count number of switch presses.

```
main(){
  ..other config..
  CONFIG_SW1();
  CONFIG_LED1();
  LED1 = 0;
  while (1) {
    if (SW1_PRESSED()) {
    //switch pressed
    //toggle LED1
    LED1 = !LED1
    }
  }
}
```

a. Incorrect, LED1 is toggled as long as the switch is pushed, which could be a long time!

```
main(){
  ..other config..
  CONFIG_SW1();
  CONFIG_LED1();
  LED1 = 0;
  while (1) {
   // wait for press, loop(1)
   while (SW1_RELEASED());
   DELAY_MS(15); //debounce
   // wait for release, loop(2)
   while (SW1_PRESSED());
   DELAY_MS(15); // debounce
   LED1 = !LED1; //toggle LED
  }
```

b. Correct, loop(1) executed while switch is not pressed. Once pressed, code becomes trapped in loop(2) until the switch is released, at which point LED1 is toggled.

Additional Code and Devices …

Additional Code and Devices

- Chapter 8 contains excellent material that you should consider using when designing your devices ...
  - State Machine for LED toggling
  - LED/switch I/O problem
  - Interfacing to an LCD Module
    - Contains ...
      - Schematic
      - LCD commands
      - LCD Code examples

Summary ...

Summary

- General Purpose Input / Output (GPIO) port usage of ...

  » PORTA, PORTB

- Weak pull-ups of PORTB
- Schmitt Trigger
- Tri-state buffers
- Open-drain output
- Writing C code for flashing and LED and LED/Switch IO

Lab

Peer Review of Software ...

Peer Review of Software Developed

- Next week!!!
  - Be prepared
    - How you wrote your code
    - Problems encountered
    - Questions you need solved

Lab #1 ...

Lab #1

- Initial breadboard wiring
- Initial processor start-up
- Initial processor software development
  - Flashing LED1
  - Switch interface

Next Class

Next Class Topics

- Lab #1 continued

Homework

Homework

- Read ...
  - Material covered in today's lecture (may want to re-read) ...
    - Chapter 8, pages 287 – 312
  - Material for lecture in two weeks ...
    - Chapter 9, pages 317 - 362
- Work on Lab#1
  - Breadboard wiring
  - Code development

Time to start the lab …

Lab

- Start Lab #1 ...

References

References

1. None